Micro Estación Metereológica

Luis Rodolfo Torres Contreras and José Carlos García Montes

Universidad Autónoma de Nuevo León

Facultad de Ciencias Físico Matemáticas

Junio 2019

Índice general

1. Resumen **2**

2. Antecedentes **3**

3. Objetivos **4**

4. Procedimientos **5**
4.1. Esp8266 5
4.2. Sensor MQ7(CO) 6
4.3. Sensor MQ131(O3) 6
4.4. Sensor Temperatura/Humedad DHT11 7
4.5. Sensor de polvo Sharp GP2Y1014AU0F 8
4.6. MCP3008 Convertidor A/D 8
4.7. Software 9

5. Conclusiones **11**

Capítulo 1

Resumen

En este proyecto se construirá una micro estación meteorológica capaz de poder medir la calidad del aire, con el fin de poder informar a toda la población universitaria sobre la calidad del aire del campus universitario.

Además de proporcionar la información a los estudiantes, nos propondremos a crear un sistema escalable, capaz de crecer en alcance. Brindaremos una plataforma que pueda ser utilizada por cualquier facultad de la Universidad Autónoma de Nuevo León para colocar su propia micro estación meteorológica y así brindarle información de utilidad a todos sus alumnos.

Capítulo 2

Antecedentes

La base de nuestro proyecto se logró gracias a una investigación previa "Design of a Smart Sensor Network System for Real-Time Air Quality Monitoring on Green Roof", donde la principal premisa consistía en reducir la contaminación con la ayuda de áreas verdes en los techos de los edificios. Para poder medir el impacto de esta implementación optaron por diseñar una estación meteorológica para poder medir la calidad del aire en dicha zona.

Lo novedoso de este proyecto es que podían comunicarse con la estación a través de internet con un protocolo enfocado al IoT, el protocolo MQTT es el ideal porque es fácil de implementar y es ligero en cuanto a ancho de banda, la mayoría de los proveedores de servicios en la nube usan este protocolo por lo antes mencionado.

Y de esta investigación fue donde nos nació la idea de diseñar una estación meteorológica para medir la calidad del aire dentro de nuestra institución, la "Universidad Autónoma de Nuevo León".

Teniendo como base estos principios partimos a realizar nuestro proyecto, el cual consiste en obtener mediciones de calidad del aire para evaluar nuestro entorno universitario y tomar decisiones de acuerdo con los resultados obtenidos y hacer implementaciones para la mejora del entorno universitario.

De acuerdo con los "Límites establecidos por el gobierno mexicano y el gobierno estadounidense con respecto a la emisión de partículas contaminantes", hemos obtenido las unidades de medición para estar dentro de este reglamento. Donde para las PM10 y PM2.5 usaremos la unidad de medición $\mu g/m3$, para el O3 utilizaremos ppm, para mq7 usaremos ppm, para la temperatura se estará midiendo en grados centígrados y la humedad será medida usando porcentajes.

Capítulo 3

Objetivos

El objetivo principal del proyecto es establecer un registro histórico de las condiciones climatológicas relacionadas con la contaminación en el campus universitario de la Universidad Autónoma de Nuevo León. El sistema también tiene como objetivo proporcionar la facilidad para que cada facultad de la universidad integre las lecturas de sus propias estaciones meteorológicas y así poder construir un mapa detallado de las condiciones ambientales de todo el campus.

El registro histórico podrá ser consultado por los estudiantes desde una aplicación móvil, esto concientizará acerca de la situación actual ambiental de la universidad además de proporcionar una herramienta para que los estudiantes tomen medidas necesarias para evitar la exposición a contaminantes que puedan dañar su salud.

Tener un registro histórico, además de ayudarnos a saber cómo va empeorando la calidad del aire de la zona del campus también nos servirá para establecer mecanismos de predicción en la condición climática del mismo.

Capítulo 4

Procedimientos

El proyecto incluye una micro estación meteorológica, un sistema encargado de recibir la información enviada por la estación meteorológica y por últimos, las interfaces de usuario que serán tanto web como móviles.

4.1. Esp8266

El microprocesador usado es el Esp8266, es un microcontrolador muy económico y de muy buena calidad, tiene integrado un circuito WiFi y puerto USB. Podemos programar el chip en el IDE Arduino utilizando el lenguaje de programación C. También cuenta con un chip que tiene varios pines para conectar sensores (digital, 1 análogo, I2c, SPI, MISO...), todo esto en un circuito de 25mm x 48mm. Esto significa que se pueden conectar sensores de casi cualquier tipo para controlarlos y comunicarnos con el servidor mediante wifi.

Pasos para la instalación y configuración

1. IDE Arduino Y Placa ESP8266

   a) Instalar IDE Arduino: `https://www.arduino.cc/`

   b) Agregar la placa ESP8266: `https://www.luisllamas.es/programar-esp8266-con-el-ide`

2. Sensores - Se configuraron sensores de Temperatura/Humedad, Densidad de Polvo, Ozono y CO.

3. Circuito - Para el circuito se tuvo que incorporar un convertidor de señal análoga a digital porque el ESP8266 solo tiene una entrada analógica el circuito mencionado es el MCP3008 compatible con el protocolo SPI.

4.2. Sensor MQ7(CO)

Para hacer funcionar este sensor se buscó una librería en GitHub: `https://github.com/swatish17/MQ7-Library`

Lo que necesitamos para hacer funcionar el sensor es agregar estas dos líneas en la cabecera del programa:

```
#include "MQ7.h"
MQ7 mq7(A0,5.0);
```

La primera línea consta de la librería esto es para poder acceder a las y métodos de ella. En la segunda línea se instancia un objeto de la clase MQ7, por parámetros se le pasa el puerto análogo y el voltaje. Y en el void loop agregamos esta instrucción para leer los valores en partes por millón.

```
float ppm=mq7.getPPM();
```

4.3. Sensor MQ131(O3)

Este sensor fue un poco de mas trabajo porque no encontré una librería así que configuré el sensor directo con el datasheet. Navegando por la hoja de datos encontré una grafica de sensibilidad del sensor donde:

- Ro es un valor que se obtiene en un laboratorio.
- Rs es la resistencia interna del sensor.
- Y ppm es la unidad que lee el sensor.

Para poder configurar nuestro sensor tenemos que estimar los valores de la recta de ozono para encontrar la ecuación, se puede hacer en Excel, esto es tomar tantos puntos de la recta y graficarlos, después tenemos que marcar una línea de tendencia potencial y hacer mostrar la ecuación. Estos fueron mis resultados:

Lo siguiente que nos queda es pasar al código:

```
int adc_MQ = adc.readADC(0);
float voltaje = adc_MQ * (3.0 / 1023.0);
float Rs=(1000*((5-voltaje) /voltaje));
double ozono=39.707*pow(Rs/339.5, -1.497);
```

La primera línea de código consiste en leer los datos de los sensores. La segunda línea tenemos que pasar esa lectura voltaje. En la tercera línea obtenemos el valor de Rs.

Ahora solo nos queda obtener Ro para esto como no tengo un ambiente controlado de CO y no puedo hacer que mi sensor aumente en voltaje lo que hice fue conectar un potenciómetro y hacer una lectura de 1023 para sacar el valor máximo de $Rs = 679$, después de esto solo sustituí el valor máximo con el ultimo dato de mi grafica y obtuve $Ro/Rs = 0{,}5$.

El ultimo dato que se obtuvo de la grafica fue $Ro/679 = 0{,}5$. Despejando $Ro = 679 * {,}5 = 339{,}5$

Así fue como se obtuvo el valor en partes por millón.

4.4. Sensor Temperatura/Humedad DHT11

A este sensor le incorporan una librería específica para poder usarlo, pueden buscarla directamente en el gestor de librerías de Arduino e instalar la ultima versión.

Para poder programarlo se necesita:

```
#include <DHT.h>
#define DHTPIN 0 //DIGITAL 3
#define DHTTYPE DHT11
DHT dht(DHTPIN, DHTTYPE);
```

La primera línea de código agrega la librería en la cabecera del programa, después se define el pin digital al cual será conectado en la placa. Posteriormente instanciar un objeto de la clase DHT y pasar por parámetro el pin y después el tipo del sensor.

Para inicializar el sensor tenemos que agregar una instrucción en el void setup:

```
dht.begin();
```

Para poder obtener los valores lo que haces en la función void loop es agregar estas dos líneas:

```
float temp=dht.readTemperature();//Resultado Final
float humedad=dht.readHumidity();//Resultado Final
```

Nos guardara el valor de la temperatura en temp y el de la humedad en la variable humedad.

Para poder visualizar los datos en el puerto serie usamos la instrucción Serial.print(), pasando por parámetro el nombre de la variable a imprimir.

4.5. Sensor de polvo Sharp GP2Y1014AU0F

Este sensor usa el principio de dispersión de la luz, de manera que tiene un emisor led y un fotodetector que se oponen entre si formando un ángulo dentro de la capsula del sensor que tiene un orificio por el cual pasa el polvo. De manera que cuando hay polvo dentro de la cavidad del sensor causa que la luz del emisor se disperse hacia el fotodetector, cuanto mas polvo haya en la cavidad mayor será la intensidad de la luz dispersada, el sensor emite un valor de voltaje en base a la luz dispersada y podemos detectar la densidad de polvo utilizando una relación lineal.

En otras palabras, el polvo entra por la cavidad mientras que la luz emitida se dispersa hacia el fotodetector y este emite un voltaje con relación a la intensidad de la luz.

Para una explicación más detallada del código para este sensor se deja el siguiente link de GitHub para consulta: `https://github.com/sharpsensoruser/sharp-sensor-demos`

4.6. MCP3008 Convertidor A/D

El mcp3008 es un convertidor de analogico a digital, se utilizo porque tenemos 3 sensores con entrada analagica(MQ131,MQ7, Sensor de Polvo) como nuestra placa ESP8266 solo tiene una entrada analogica A0 tuvimos que encontrar un circuito integrado compatible con nuestra placa.

Gracias a que nuestra placa y el mcp3008 son compatibles con el protocolo de comunicación SPI(Serial Peripheral Interface), pudimos obtener los datos de nuestros sensores.

El protocolo SPI utiliza 4 señales para comunicarse, sincronizar y poder crear las 8 entradas análogas:

SCLK (Clock): Es el pulso que marca la sincronización. Con cada pulso de este reloj, se lee o se envía un bit.

MOSI (Master Output Slave Input): Salida de datos del Master y entrada de datos al Esclavo.

MISO (Master Input Slave Output): Salida de datos del Esclavo y entrada al Master.

SS/Select: Para seleccionar un Esclavo, o para que el Master le diga al Esclavo que se active. También llamada SSTE.

Para programar el microcontrolador se utilizó la librería de Adafruit mcp3008 que se puede instalar usando el gestor de librerías de arduino te incluye dos códigos de ejemplo pero modificamos uno para poder implementarlo en nuestro sistema.

En la primer línea se incluye la librería con la cual se está trabajando MCP3008. Después tenemos que crear un objeto de la clase Adafruit _MCP3008 al cual llamamos adc. Posteriormente, definimos nuestros pines y se especifica a que pines van conectados de la placa.

En el void setup asignamos la velocidad de transmisión de los datos por el puerto serial que es de 9600 baudios. También iniciamos el mcp3008 y le decimos porqué puertos va a recibir las 4 señales. Y en el void loop creamos una variable entera para almacenar el valor de la entrada analógica con la instrucción adc_MQ=adc.readADC(0); que indica que estaremos leyendo datos del canal 0 del mcp3008.

4.7. Software

El protocolo usado para la transferencia de los datos es MQTT. Este protocolo nos ofrece ventajas ideales para su uso en aplicaciones de IoT ya que nos permite trabajar con un ancho de banda limitado y alta latencia. La arquitectura del protocolo nos permite tener sensores que publiquen los datos en un Broker el cuál a su vez puede enviar la información a los clientes que estén subscritos a él. Así, podemos asegurarnos que toda la información llegará a los clientes finales de una manera sencilla.

El Broker que usaremos es Mosquitto MQTT ya que es de fácil instalación en sistemas Linux y nos permite crear una lista de acceso que nos permite

controlar que dispositivos pueden enviar y recibir datos del mismo. Todas estas configuraciones se llevaron a cabo en los archivos de configuración después de la instalación del Broker.

Al Broker tendremos subscrito un sistema cliente encargado de guardar toda la información histórica de los sensores. Este cliente esta escrito en el lenguaje de programación Python y usa la librería Paho MQTT que nos permite hacer la subscripción al Broker.

Este programa está activo en un loop infinito controlado proporcionado por la librería Paho el cual está constantemente escuchando todos los mensajes que lleguen desde el Broker. Después de recibir el mensaje, el código identifica la facultad y la micro estación de procedencia del mismo mediante la estructura del topic con el que el Broker compuso el mensaje. Al momento de identificar estos datos, se hace uso de la librería PyMongo para poder guardar en la base de datos no relacional un nuevo documento en donde se describa el mensaje enviado, la micro estación que emitió el mensaje, el tipo de cualidad atmosférica que se midió, la facultad dueña de la micro estación y la hora en la que se registro la medición.

En nuestro caso escogimos usar la base de datos MongoBD por su condición open source. Para los datos específicos de las estaciones meteorológicas como su ultima fecha de mantenimiento, su propietario o la información específica de sus sensores, se utilizará la base de datos relacional MySQL.

Se desarrolló una aplicación web que servirá como interfaz para poder vigilar el estado de los diferentes sensores del campus universitario así como también las mediciones que estos vayan generando, la aplicación web cuenta con un backend hecho en Python con el framework Flask y un frontend hecho con JavaScript y VueJS. Para no batallar con la creación de la aplicación para varias plataformas, se decidió usar la tecnología Xamarin de Microsoft que nos permitirá terminar de desarrollar una aplicación capaz de correr en cualquier sistema operativo movil, la aplicación simplemente servirá para poder observar toda la información histórica producida por los sensores.

Capítulo 5

Conclusiones

Se espera que con este proyecto se pueda tener una certeza de que es lo que está pasando en nuestro entorno, que estamos respirando tener información reciente y refutable de unas de las partículas más comunes que despiden las industrias y al mismo tiempo dañinas para nuestro organismo saber hasta qué punto es peligroso y que medidas debemos de tomar tanto en el sector empresarial como en el comunitario, este proyecto va dirigido a las industrias que no son socialmente responsables o no se están haciendo cargo del daño ambiental que están ocasionando así como también al resto de la población que convocamos a exhortar a las empresas que no estén tomando medidas para reducir las partículas dañinas al medio ambiente y por ultimo al estado para decretar leyes más estrictas sobre el control de las emisiones de las empresas